CLAIMS

What is claimed is:

1   1.   A processor, comprising:

2   a first exception handler that receives and handles critical excepted instructions; and

3   a second exception handler that receives and handles non-critical excepted instructions.

1   2.   The processor of claim 1, wherein the critical excepted instructions comprise exceptions

2   that are performance critical.

1   3.   The processor of claim 1, wherein the non-critical excepted instructions comprise

2   exceptions that are not performance critical.

1   4.   The processor of claim 2, wherein the critical excepted instructions include branch

2   mispredictions.

1   5.   The processor of claim 2, wherein the critical excepted instructions include load/store traps.

1   6.   The processor of claim 2, wherein the critical excepted instructions include jump

2   mispredictions.

1   7.   The processor of claim 3, wherein the non-critical excepted instructions include illegal

2   instructions.

1   8.   The processor of claim 3, wherein the non-critical excepted instructions include cache

2   parity errors.

1   9.   The processor of claim 3, wherein the non-critical excepted instructions include invalid

2   instructions.

1   10.   The processor of claim 3, wherein the excepted instructions include arithmetic overflows.

1   11.   The processor of claim 1, wherein the first exception handler operates speculatively.

1   12.   The processor of claim 1, wherein the first exception handler causes critical excepted

2   instructions to be resolved on a speculative basis, even though the excepted instruction may not be

3   in an actual program path.

1   13.   The processor of claim 1, wherein the second exception handler operates non-speculatively.

1   14.   The processor of claim 1, wherein the second exception handler causes non-critical

2   excepted instructions to be resolved only when it is certain that the excepted instruction is in an

3   executing program.

1   15.   The processor of claim 1, further comprising a plurality of pipelines with multiple stages,

2   and wherein excepted instructions may arise in one or more of the pipeline stages.

1   16.   The processor of claim 15, wherein excepted instructions arising from said one or more

2   pipeline stages is routed to the first exception handler or second exception handler based on a

3   predetermined criteria.

1   17.   The processor as in claim 16, wherein the predetermined performance criteria relates to

2   performance of the processor.

1   18.   An exception handler for a processor that resolves excepted instructions , comprising:

2   a speculative exception handler that receives critical excepted instructions and resolves said

3   critical excepted instructions on a speculative basis; and

4   a non-speculative exception handler that receives non-critical excepted instructions and

5   resolves said non-critical excepted instructions on a non-speculative basis.

1   19.   The exception handler of claim 18, wherein the speculative exception handler causes

2   critical excepted instructions to be expeditiously resolved even though the critical excepted

3   instruction may not be in an actual path of an executing program.

1 20. The exception handler of claim 18, wherein the non-speculative exception handler delays

2 resolution of said non-critical excepted instructions until it is certain that said non-critical excepted

3 instruction lies in an actual path of an executing program.

1 21. A processor, comprising:

2   at least one pipeline with a plurality of stages;

3   an algorithm for detecting non-executable instructions in said at least one pipeline, wherein

4 said algorithm generates a command that identifies the non-executable instruction and identifies a

5 reason that the non-executable instruction will not execute;

6   a speculative exception handler that receives said command for any non-executable

7 instructions that are critical to processor performance; and

8   a non-speculative exception handler that receives said command for any non-executable

9 instructions that are not critical to processor performance.

1 22. The processor claim 21, wherein the speculative exception handler expeditiously resolves

2 critical non-executable instructions even though the critical non-executable instruction may not be

3 in an actual path of an executing program.

1 23. The processor of claim 21, wherein the non-speculative exception handler delays resolution

2 of said non-critical non-executable instructions until it is certain that a non-critical non-executable

3 instruction lies in an actual path of an executing program.

1 24.   The processor as in claim 22, wherein said speculative exception handler includes logic for

2 resolving critical non-executable instructions.

1 25.   The processor as in claim 23, wherein said non-speculative exception handler includes

2 logic for resolving non-critical non-executable instructions.

1 26.   A method of handling exceptions in a processor during the execution of a program,

2 comprising the acts of:

3   detecting an exception in one or more stages of one or more pipelines;

4   identifying if the exception is critical to the performance of the processor;

5   routing critical exceptions to a first exception handler;

6   routing all non-critical exceptions to a second exception handler; and

7   expeditiously resolving critical exceptions.

1 27.   The method of claim 26, wherein the critical exceptions are expeditiously resolved by

2 handling the critical exceptions speculatively.

1 28.   The method of claim 26, wherein the critical exceptions are expeditiously resolved even

2 though the critical exception may not be in an actual path of the program.

1   29.   The method of claim 26 wherein a plurality of pipelines are provided for executing

2   multiple program instructions in parallel, and wherein exceptions are detected in each stage of each

3   pipeline.

1   30.   The method of claim 26, wherein the second exception handler delays resolution of said

2   non-critical exceptions until it is certain that a non-critical exception lies in an actual path of the

3   program.

1   31.   A method of handling exceptions in a processor during the execution of a program,

2   comprising the acts of:

3   detecting an exception and identifying if the exception is critical or non-critical to the

4   processor performance;

5   routing critical exceptions to a speculative exception handler; and

6   routing all non-critical exceptions to a non-speculative exception handler.

1   32.   The method of claim 31, wherein the critical exceptions are resolved speculatively, even

2   though the critical exception may not be in an actual path of the program.

1   33.   The method of claim 31, wherein the non-speculative exception handler delays resolution

2   of said non-critical exceptions until it is certain that a non-critical exception lies in an actual path of

3   the program.

1   34.   The method of claim 33, wherein the exceptions from the non-speculative exception

2   handler are selected for resolution prior to the exceptions from the speculative exception handler.